TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

TVORBA ”OFFLINE” WEBOVÝCH APLIKACÍ: SPORTOVNÍ ČASOMÍRA

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Michal Mužíček**
Vedoucí práce: Ing. Jana Vitvarová, Ph.D.

Liberec 2014

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

DEVELOPING "OFFLINE" WEB APPLICATIONS: SPORT TIMER

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Michal Mužíček**
Supervisor: Ing. Jana Vitvarová, Ph.D.

Liberec 2014

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michal Mužíček**
Osobní číslo:	**M11000109**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Tvorba "offline" webových aplikací: Sportovní časomíra**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s prvky HTML5, které umožňují tvorbu tzv. "offline" webových aplikací.
2. Navrhněte a implementujte "offline" webovou aplikaci, která umožní v terénu měřit čas závodníků a bude poskytovat další funkce potřebné pro spravování sportovních závodů.
3. Aplikaci navrhněte tak, aby byla případně použitelná i v mobilním telefonu.
4. Aplikaci otestujte v terénu.
5. Vytvořte dokumentaci k aplikaci a vhodně okomentujte zdrojový kód.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **30–40 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Connolly T., Begg C., Strachan A.: Database systems, Addison-Wesley, 1999**

[2] **Pilgrim M.: Dive into HTML5 dostupné na: http://diveintohtml5.info/index.html**

[3] **HTML5 Features Offline dostupné na: http://www.html5rocks.com/en/features/offline**

Vedoucí bakalářské práce: **Ing. Jana Vitvarová, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: **10. října 2013**

Termín odevzdání bakalářské práce: **16. května 2014**

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci - 14 -

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2013

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15. 5. 2014

Podpis:

Poděkování

Rád bych na tomto místě poděkoval své vedoucí, Ing. Janě Vitvarové, PhD., za její trpělivost a veškerou její pomoc při tvorbě této bakalářské práce. Zároveň bych rád poděkoval svému bratrovi, který mi nabídnul zadání této práce, ve chvíli kdy jsem si nebyl jistý jaké téma bych si měl vybrat.

Abstrakt

Práce se zabývá tvorbou offline webové aplikace s použitím prvků HTML5 pro zpracovávání výsledků sportovních závodů. Pojem offline webová aplikace znamená, že aplikace je schopná pracovat i v případě nedostupného připojení k internetu. V tomto případě neukládá svá data přímo na server, ale je schopna si je dočasně uložit na lokální úložiště a po obnovení internetového připojení se uložená data přenesou na server. Práce demonstruje použití webových prvků HTML, PHP, CSS a JavaScript pro ukládání a zpracování dat potřebných pro vytvoření výsledků sportovního závodu bez nutnosti připojení k internetu.

Klíčová slova

HTML5, offline, sport, časomíra, webová aplikace

Abstract

This documentation describes a development of an offline web aplication that uses HTML5 elements for processing the results of sport races. The term offline web application means that application is capable of working even without internet connection. In such case the application doesn't save its data directly to a server, but is capable of saving it to local storage and when the internet connection is established, the saved data are transferred to the server. The application shows the use of web elements HTML, PHP, CSS and JavaScript to save and process the results of a sport race without the need for internet connection.

Keywords

HTML5, offline, sport, timer, web applications

Obsah

Seznam obrázků

Seznam ukázek kódu

Seznam zkratek

HTML5 Hyper Text Markup Language 5 10

HTML HyperText Markup Language 11

HTTP Hypertext Transfer Protocol 11

PHP Hypertext Preprocessor 11

Úvod

Jelikož často pomáhám při organizování amatérských závodů a vždy musíme řešit časomíru, tak mým cílem bylo vytvořit nástroj, který nebude finančně náročný (oproti čipům, které jsou nákladné) a zároveň usnadní tento proces (oproti ručnímu zaznamenávání). Časomíra je jeden z nejdůležitějších aspektů sportovního závodu, jelikož se o ni opírá tvorba výsledků a tedy zhodnocení celého závodu. Cílem bakalářské práce je vytvořit webovou aplikaci, která je schopna pracovat bez připojení k internetu, jejíž cílem je zpracovávání časů a výsledků.

Práce se zabývá tvorbou webové aplikace s použitím nových prvků, které přináší Hyper Text Markup Language 5 (HTML5) (viz. HTML5: Up and Running [1]). Zaměřuje se na využití tzv. „offline“ ukládání dat (tedy pokud nejste připojeni k internetu), které zmíněný standard poskytuje. Toto je z důvodu omezené možnosti připojení k internetu v terénu, kde se závody většinou pořádají, zatímco aplikace má umět následné automatické nahrání na web. Taková to aplikace je pak vhodná i pro použití v chytrých telefonech, které lze jednoduše vzít do terénu.

Úvod do HTML5

HTML je rozšířený jazyk pro psaní kostry webových stránek. Jeho nejnovější verzí je HTML5, jehož prvky umožňují tvorbu tzv. „offline“ webových aplikací. Uvedený standard popisuje jak takovéto prvky vypadají, jak se s nimi pracuje a jak konkrétně umožňují „offline“ ukládání dat. A ve spojení s PHP, které umožňuje automatické generování obsahu webových stránek, již máme možnost vytvořit offline webovou aplikaci.

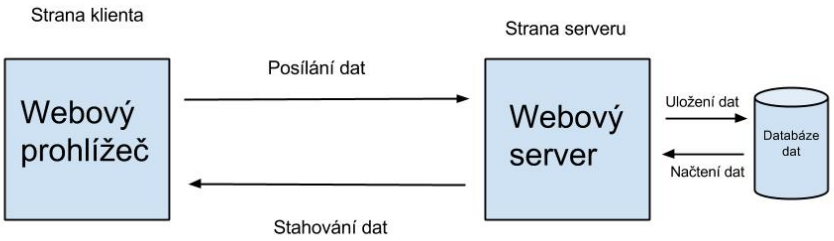
1 Tvorba offline webových aplikací

1.1 Webová aplikace

Základní architektura webových stránek je typu statický obsah, třívrstvá dynamická aplikace nebo n-vrstvá dynamická aplikace.

Pod pojmem webová aplikace rozumíme dynamickou aplikaci, což je program, který byl vytvořen pro použití ve webovém prohlížeči. Jak je uvedeno v [3], webové aplikace jsou klasické webové stránky, které však dokáží zvládnout práci s dynamickými funkcemi podobně jako desktopové aplikace nainstalované na počítači. S tím rozdílem, že webové aplikace nemusíte instalovat na počítač a k jejich spuštění potřebujete pouze webový prohlížeč. Pracuje se s jazyky pro tvorbu webových stránek HyperText Markup Language (HTML),Hypertext Preprocessor (PHP) a Java Script, které zajistí výše zmiňované funkce. Většina těchto funkci je zpracována na straně serveru.

Webové aplikace se běžně vytváří na základě 3 vrstev. První je prezentační, kterou nejběžněji zastupuje webový prohlížeč. Druhá je aplikační vrstva, kam patří nástroje pro dynamické generování stránek (např. PHP) a poslední je datová, kterou představuje databáze. Tato struktura pak funguje tak, že webový prohlížeč posílá požadavky druhé vrstvě, jež na ně odpovídá prostřednictvím dotazů z databáze a generováním uživatelského rozhraní.



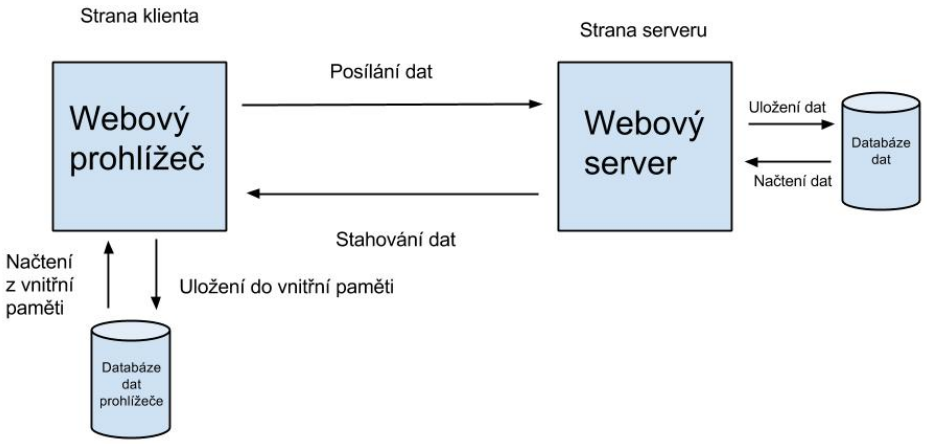
Obrázek 1: Webová aplikace

Na obrázku č. 1 je vidět schéma komunikace mezi klientem (uživatelský webový prohlížeč) a serverem (např. Apache) pomocí protokolu Hypertext Transfer Protocol (HTTP). Uživatel si vyžádá např. webovou stránku a tento požadavek (HTTP Request) pošle serveru. Server požadavek přijme, podívá se do své databáze a pokud nalezne požadovaná data, pošle pozitivní odpověď (HTTP Response) zpátky klientovi. V případě, že se jedná o stránku s dynamickým obsahem, je tato stránka nejdříve vygenerována pomocí PHP kódu. V odpovědi od serveru je pak umístěná webová stránka skládají se z HTML, CSS a JavaScript kódu, který prohlížeč zpracuje a zobrazí uživateli.

Webová aplikace pak představuje většinou komplexnější operace, jako například objednávání zboží na internetovém obchodě. Uživatel si zvolí svůj nákup a potvrdí objednávku. Přičemž tato objednávka se pošle na server, kde se uloží do databáze pro pozdější zpracování.

1.2 „Offline“ webová aplikace

„Offline“ webová aplikace, je taková aplikace, která ke svému chodu nepotřebuje nutně připojení k serveru. Takováto aplikace je schopna pracovat dále i v offline režimu a při pozdějším připojení se tzv. synchronizuje (tj. obsah na serveru se aktualizuje o data vytvořené v režimu offline). Například e-maily, které jste napsali při výpadku internetu, se uloží do vnitřní databáze dokud se neobnoví spojení a poté se automaticky uloží na server.



Obrázek 2: „Offline“ webová aplikace

Jak je z obrázku vidět, oproti klasické webové aplikaci, přibyla vnitřní databáze, do které si aplikace ukládá veškerá „offline“ data. Jinými slovy data z chodu aplikace, když není k dispozici internetové připojení. To umožňuje speciální záznam v hlavičce HTML stránky ve spojení s manifestem (viz. níže).

Jak jsem již uvedl výše, nejjednodušší příklad „offline“ je psaní e-mailů (například Gmail, který tento režim podporuje).Tento režim pak funguje i tak, že když při rozepsaném e-mailu a vypadne spojení, tak Gmail začne ukládat vaši práci na místní úložiště (pod prací se rozumí napsaní zpráv, odeslání zpráv, mazání zpráv atd.). A jakmile se obnoví spojení tak se provedené akce zrealizují, tedy zprávy se odešlou apod. Ovšem podmínkou je mít offline režim zapnutý. Bez něj vám Gmail pouze napíše, že některé prvky nebudou pracovat správně, jelikož vyžadují režim online. A pokud se takto pokusíte na Gmail přistoupit tak se vám aplikace ani nezobrazí.

Samotná kostra aplikace se však tolik neliší. Hlavním rozdílem je údaj v HTML hlavičce (viz. ukázka kódu 1). Ten prohlížeči dá pokyn, že v případě nepodařeného spojení má hledat alternativu a přídavný soubor zvaný manifest (viz. ukázka kódu 2). Manifest uchovává informaci o všech souborech, které se mají ukládat do vnitřní paměti.

Ukázka kódu 1: Změna v hlavičce HTML

```
<html manifest="application.appcache">
```

Ukázka kódu 2: Ukázka těla manifestu

```
CACHE MANIFEST
style.css
/main/home
/main/app.js
http://img.example.com/logo.png
```

Můžeme vidět, že do tohoto manifestu spadají 3 soubory a 1 adresář. Ukázka z webových stránek Interval.cz [4].

1.2.1 Problematika zjištění připojení

Pro moji práci je daná problematika stěžejní, jelikož rozhodování zda aplikace má přístup k serveru či nikoli je základem mé aplikace. Ač se to zdá na první pohled jasné, komunita prohlížečů se na uvedené problematice rozchází [7]. Řešil jsem to u dvou podle mě nejrozšířenějších a zároveň nejvyspělejších prohlížečů.

Firefox tento stav definuje jako práci v „offline“ režimu, čímž nastaví speciální atribut navigator.onLine na hodnotu false. Uvedená varianta byla pro mé účely nepoužitelná, jelikož se prohlížeč musí manuálně přepnout do tohoto režimu. Navíc při testování jsem zjistil, že i když se odpojím od sítě, tak je tento atribut nastaven pořád na true.

Chrome nemá žádný „offline“ režim, jelikož on nastaví zmíněný atribut při ztrátě spojení, ovšem zde zase nastává kolize dvou termínů a to internetové připojení (tedy připojení k internetu) a síťové připojení (připojení k místní síti). Pro mé účely potřebuji zjistit internetové připojení ale Chrome je schopen zjistit pouze síťové připojení, takže ani tento postup není přijatelný.

Nakonec jsem zvolil vlastní cestu. Na serveru si připravím velmi malý soubor, který se vždy pokusím stáhnout. Stav „offline“ pak definuji tím, že se stáhnutí nepodařilo.

1.2.2 Jednoduchý příklad na začátek

Když jsem s mojí prací začínal, vybral jsem si jednoduchou aplikaci pro vyzkoušení konceptu. Na stránkách FT Labs [5] byl vhodný příklad aplikace typu „RSS feed“ (informační kanál, kde se často přidávají nové zprávy). V jejich příkladu je tvorba této aplikace brána pomalu a postupně, takže byla ideální jako začátek.

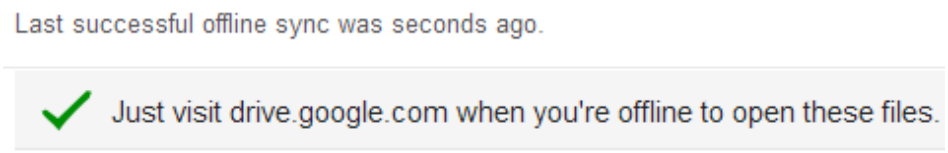
Nejdůležitější část je soubor zvaný manifest, jelikož přímo ovlivňuje jaké soubory budou přístupné v offline části aplikace.

Ukázka kódu 3: Tělo manifestu v příkladu

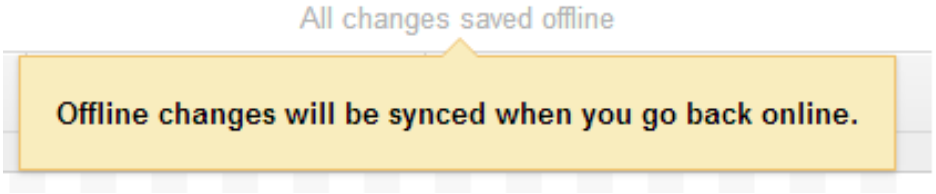
```
<?php
header("Content-Type: text/cache-manifest");
?>
CACHE MANIFEST
# 2012-07-14 v2
jquery.min.js
/
NETWORK:
*
```

Zde můžeme vidět hlavní část. A to je „CACHE MANIFEST“, která udává, jaké soubory se mají uchovat v tzv. cache paměti (neboli vnitřní dočasná paměť). Což jsou dvě položky, a to „jquery.min.js“ (soubor poskytující funkce pro práci s Java Scriptem) a „/“ (to označuje aktuální adresář), což v tomto případě zahrnuje pouze „index.html“.

To pro nás znamená, že při prvním přístupu k této aplikaci si prohlížeč tyto soubory uloží do své paměti a pokud se někdy stane, že není dostupné spojení, tak použije právě tyto soubory ke své práci, čímž vzniká offline aplikace.



Obrázek 3: Google online



Obrázek 4: Google offline

V obrázku č. 3 je vidět aplikace Google Drive během připojení k jejich serverům (v překladu je zde napsáno, že poslední úspěšná synchronizace proběhla před několika sekundami a pokud chceme k těmto datům přistoupit bez připojení, stačí otevřít drive.google.com).

A na obrázku č. 4 můžete vidět Google Drive v režimu offline (v překladu je zde napsáno, že všechny změny jsou ukládány „offline“ a budou synchronizovány jakmile bude připojení znovu k dispozici). Tento stav se musí nejdříve nastavit (tuto možnost lze nalézt v nabídce „Více“ a „Offline“).

1.2.3 Režim ukládání dat

Úložiště v HTML5 podporuje dva režimy, a to synchronní a asynchronní.

Synchronní režim je blokující, to znamená, že dokud daná operace není vykonána, další kód se nespustí. Oproti tomu asynchronní, která je neblokující, se vykoná v pozadí a po vykonání operace se zavolá obslužná procedura (tedy co má aplikace dělat když operace skončí).

Ačkoli se může zdát synchronní režim jednodušší a příjemnější, není tomu tak. Tento režim by se měl používat co nejméně to jde. Tím, že je blokující, nastává situace kdy celý prohlížeč „zamrzne“, jelikož čeká až se operace vykoná. Klasickým příkladem je, když na stránkách kliknete na tlačítko a všechno přestane reagovat, přičemž nevíte jestli aplikace přestala úplně fungovat, nebo zda-li za chvíli opět začne.

Ukládací formát „LocalStorage“ je synchronní úložiště, měl by se tedy používat pouze u rychlých a nenáročných operací avšak je nejvíce rozšířen v rámci prohlížečů. Naproti tomu „SQLite“ je asynchronní databáze, která je mnohem efektivnější při zpracovávání velkého

množství dat a operace s nimi. Díky tomu že to je databáze, je práce s daty rychlejší (indexování položek, tedy rychlé vyhledávání). Bohužel podpora popsaného režimu je menší (z prohlížečů pro stolní počítače ji podporuje Chrome, Safari a Opera).

1.3 Databáze

Důvod proč moje aplikace používá databázi, je hlavně díky rychlosti vyhledávání ve větším objemu dat a přizpůsobenost pro asynchronní operace. Databáze umožňuje ukládat data v přehledném formátu a představuje robustní úložiště (např. možnost použití transakcí je velmi důležitá). Viz. Database Systems [2]. Důvodem rychlého vyhledávání je indexování pomocí klíčů.

1.3.1 Primární klíče

Primární klíč slouží pro unikátní identifikaci záznamu v tabulce. Znamená to tedy, že každý záznam musí mít vždy primární klíč.

Jednoduchý primární klíč Pokud se primární klíč skládá pouze z jednoho sloupce, označujeme jej jako jednoduchý. Při ukládání to znamená, že pouze tento sloupec v záznamu musí být unikátní v rámci tabulky. Nejčastější a nejjednodušší typ tohoto klíče je tzv. umělý klíč (ID), který sám o sobě nenese žádnou užitečnou informaci o ukládaném záznamu a jeho hodnota je použita pouze pro pozicování v tabulce.

Složený primární klíč Má-li klíč dva a více sloupců, nazýváme jej složený a při ukládání je kladena podmínka, že jejich kombinace musí být unikátní v rámci tabulky. Tento typ klíče se použije v případě, že některé sloupce v záznamu budou vždy jednoznačně určovat daný prvek, poté není třeba vytvářet nový sloupec pro umělý klíč. Avšak může nastat situace, kdy takovýto složený klíč by v databázové struktuře mohl zanést nejasnost (pro člověka) či dokonce zpomalit vyhledávání, v takovém případě je lépe použít umělý klíč.

Cizí klíč V tabulce se také může objevit záznam typu cizí klíč. Tímto termínem označujeme klíč, který je referencí na klíč z jiné tabulky (obvykle primární). Jeho hlavním účelem je spojit dva záznamy ze dvou tabulek k sobě. Může však zároveň sloužit i jako primární klíč.

1.3.2 Transakce

Jedná se o tzv. bezpečnou operaci z hlediska konzistence dat. Pokud se započne ukládání dat, transakce zajistí aby v případě selhání byla data vrácena do původního stavu (zde pojem konzistence dat, v případě selhání během ukládání se může změnit pouze část dat do požadovaných hodnot, přičemž většinou nemáme možnost zjistit, která data jsou správně a která ne. Říkáme, že data jsou nekonzistentní, jelikož stav dat na úložišti se neshoduje se stavem původních dat)

1.3.3 Indexování

Technologie databází využívá indexovaní klíčů pro rychlejší vyhledávání v tabulkách. V důsledku to znamená, že vybraný klíč (sloupec) je označen jako index a zkopírován do speciální vyhledávací tabulky, která právě urychlí vyhledávání, za cenu dalšího potřebného místa a výkonu pro ukládání klíčů do tabulky.

1.3.4 Webová vs. Offline databáze

Nejdůležitějším rozdílem mezi webovou a offline databází je volnost přístupu k databázi. Offline umožňuje přístup kdykoliv, kdežto webová vyžaduje přístup k internetu.

Dalším značným rozdílem je povolená velikost databáze. u webových to nastavuje poskytovatel a průměrná hodnota je 35 MB, ale u offline databází uživatel nastavuje maximální velikost ukládaných dat na disk. Do tohoto limitu se počítá všechno, tzn. i samotná aplikace včetně databáze. Tedy potenciálně je možné, že uživatel nastaví příliš malou hodnotu a aplikace nemusí fungovat tak jak by bylo očekáváno. Avšak většina uživatelů nechává tuto hodnotu na její standardní velikosti, tj. 5 MB pro každou doménu. Což je víc než dostatek pro moji aplikaci, která používá i s rezervou 200 kB.

Posledním z důležitých rozdílů je druh databáze. Webové jsou obvykle MySQL5, Oracle nebo MS SQL, kdežto offline databáze implementují SQLite. Rozdíl mezi nimi je hlavně v možných operací k použití, syntaxe bývá stejná.

Chrome podporuje WebSQL a Mozzila Firefox má indexedDB. Jedná se o měnící se prvek (může se to tedy kdykoliv změnit), ale zatím to vypadá, že v budoucnu budou všechny prohlížeče podporovat indexedDB.

1.3.5 Synchronizace dat

Při používání offline webových prvků většinou vzniká problém synchronizace dat. Jelikož aplikace pracuje s daty lokálně a na webu jsou většinou zastaralá data, je třeba tato data aktualizovat pomocí těch lokálních. Daný proces má několik možných variant.

První varianta je smazání webového obsahu a nahrání lokálního. Což je postup nejjednodušší, ale bude efektivní pouze za předpokladu, že na lokáním úložišti jsou ta nejnovější data.

Druhá varianta spočívá v porovnávání obsahu a nahrazení lišících se záznamů. Daná metoda je výpočetně náročnější, ale následné datové přesuny jsou zpravidla menší. Existuje jistě více možných postupů pro synchronizaci, ale tyto patří dle mého názoru mezi nejoptimálnější.

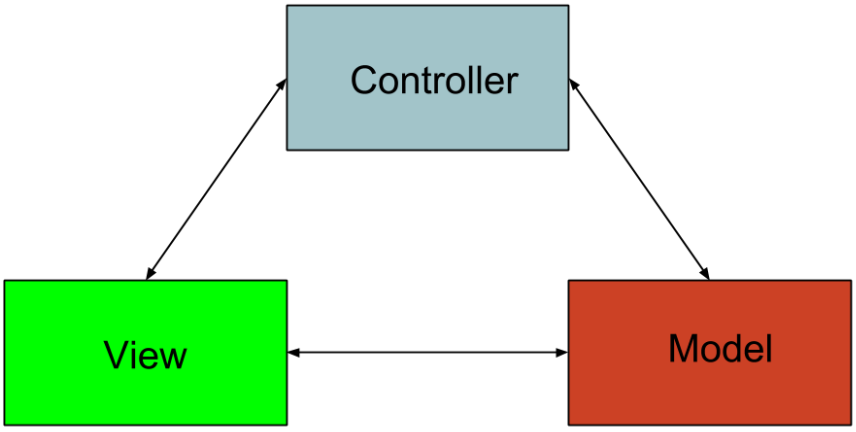
1.4 MVC architektura

Neboli Model–view–controller model, je model podle kterého se doporučuje vyvíjet aplikace, jelikož oddělením funkčních částí aplikace se nejen zjednoduší odstraňování problémů, ale také se zvýší přehlednost celého kódu. Rozděluje kód aplikace na tři části.

První je Model, který má na starosti datovou strukturu. Tedy ukládání a načítaní dat, která jsou následně zpracována (např. načtení mezd z databáze a následné počítání průměrného platu).

View modul v sobě zahrnuje kód pro zobrazování dat. Stará se tedy o to, jak požadovaná data aplikace zobrazí uživateli a jakým způsobem data od uživatele převezme. Jedná se tedy například o rozmístění prvků formuláře na stránce a jejich vybarvení.

Controller je mozek aplikace. Zpracovává veškeré funkce pro výpočet, přesun, zobrazení dat a další. Je to tedy controller, který řekne Modelu jaká data načíst z datové struktury a následně je předá View modulu, aby je zobrazil. Také přebírá uživatelská data od View modulu a následně je zpracovává.



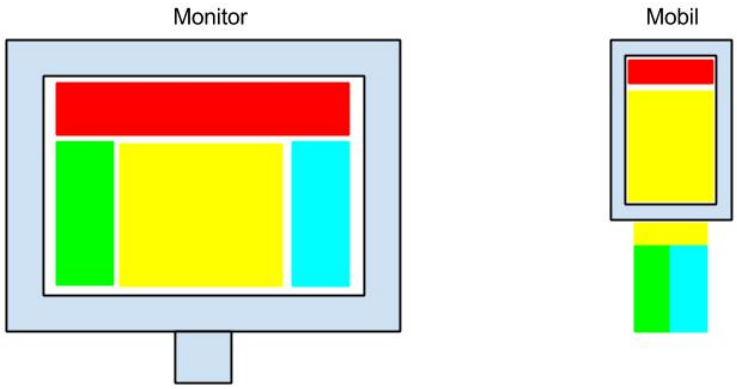
Obrázek 5: MVC model

Z obrázku č. 5 jsou patrné vztahy mezi jednotlivými moduly. Controller má přímou vazbu s Modelem, aby mohl upravovat jeho data a taktéž View má přímou vazbu, aby použitá data mohl zobrazovat. Obousměrné pozicování vazeb není podmínkou, jelikož některé aplikace mohou sloužit pouze k zobrazování dat, v tom případě by vedla vazba pouze z Controlleru do View, ale naopak už ne.

Obrázek byl inspirován webovým článkem, který napsal Borek Bernard [8].

1.5 Responsive Design

Jelikož má aplikace být kompatibilní s „chytrými“ telefony, tak je třeba zajistit správné pozicovaní prvků. Responsive design se stará o zpracování webové aplikace tak, aby se vzhled měnil podle toho na jakém zařízení je aplikace spouštěna. Například aby na klasickém monitoru byly prvky daleko od sebe, aby byla aplikace přehledná a na mobilním zařízení, aby prvky byly zmenšené a případně jinak poskládané.



Obrázek 6: Responsive Design

Na obrázku č. 6 je vidět, že ty samé prvky, jež jsou viděny najednou u monitoru, jsou zmenšeny (obvykle procentuálně) a poskládány do určité formace, v tomto případě je hlavička a tělo aplikace jako první a po posunutí obrazovky níže se dostaneme k prvkům po straně.

Jelikož dnešní informační proud směřuje směrem k menším a přenosnějším zobrazovacím zařízením (momentálně to jsou hlavně „chytré“ telefony), s ním se stává i tento koncept důležitějším. V dřívější době se tato problematika řešila pomocí duplicitních souborů, kde každý z nich byl vytvořen pro dané rozlišení, ale uvedená metoda přestává být aktuální, jelikož je mnohem efektivnější mít jeden dokument, který je schopný se přizpůsobit.

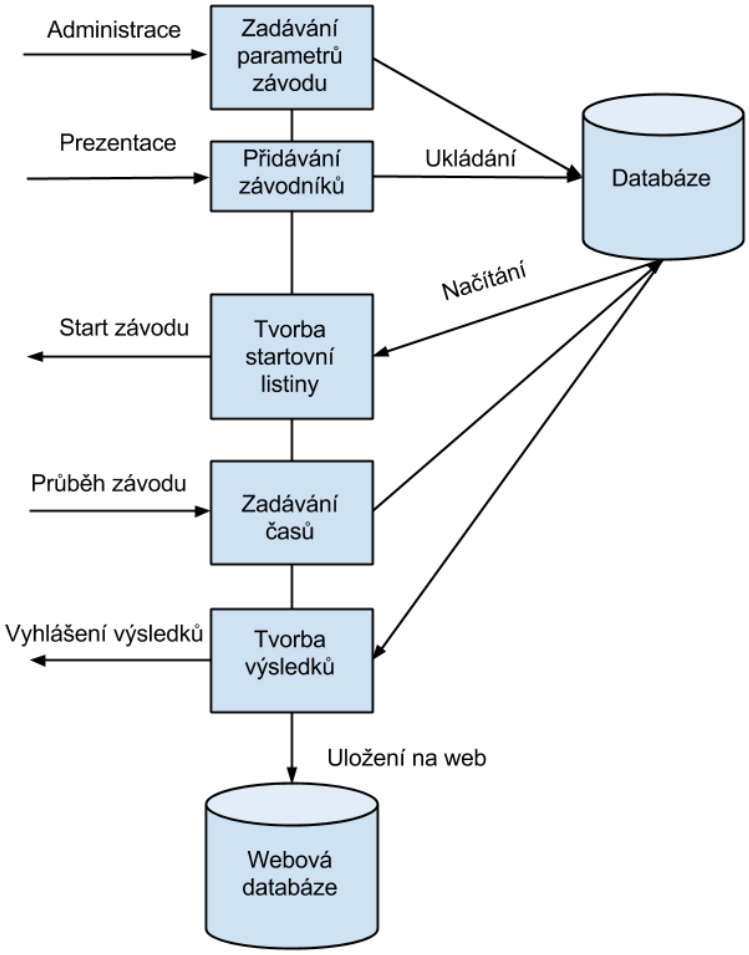
Proto důvodu vznikly tzv. frameworky (jedná se o nadstavbu, která se za vás stará o danou problematiku, zde tedy zaručuje, že se dokument přizpůsobí). Ač je vždy třeba se s nimi naučit pracovat, u větších (webových) projektů jsou pak nenahraditelné. Mezi přednější frameworky patří Foundation [6].

2 Aplikace Sportovní časomíra

2.1 Funkční analýza

Aplikace musí nejdříve získat informace o závodě, aby byla schopná jednotlivé závody rozlišit. Následně data o závodnicích, jelikož je nutné ke každému startovnímu číslu přiřadit patřičné jméno a kategorii. A nakonec samotné časy, podle kterých se budou generovat výsledky.

2.1.1 Diagram závodu

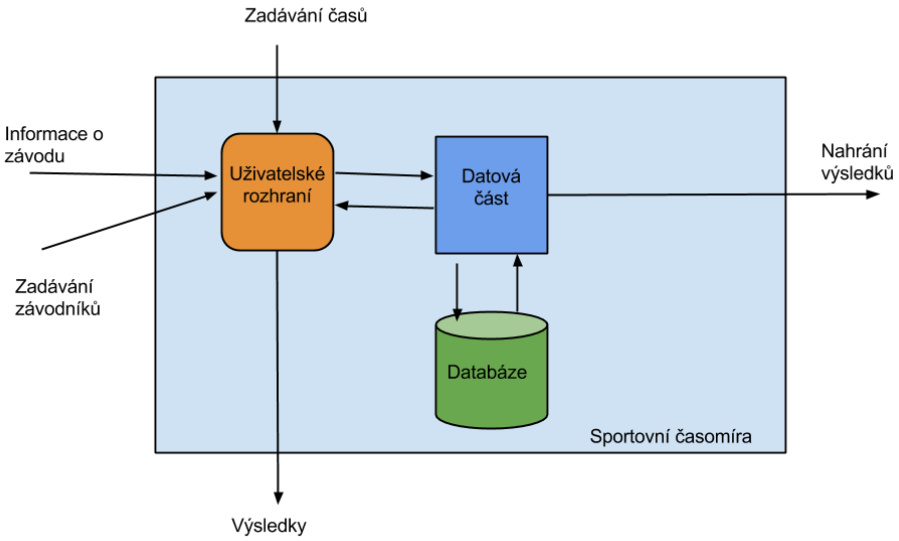


Obrázek 7: Diagram závodu

Na obrázku č. 7 jsou zobrazeny fáze závodu, jak jdou po sobě.

Během prezentace závodníci zaregistrují svoje jméno, věk a případně klub, který reprezentují a dostanou unikátní startovní číslo. Tyto údaje se uloží do databáze. Po skončení prezentace se vytiskne startovní listina (hlavně pro rozhodčí u startu a pro případného komentátora), čímž může samotný závod začít. V průběhu závodu účastníci projíždějí vyhrazeným prostorem do dalších kol, kde časomíra zaznamenává jednotlivé časy podle startovních čísel. Po skončení závodu časomíra sestaví výsledky, podle kterých se může závod vyhlásit a následně uloží získaná data na webovou databázi.

2.1.2 Funkční diagram



Obrázek 8: Funkční diagram

Na obrázku č. 8 jsou vidět základní funkce aplikace.

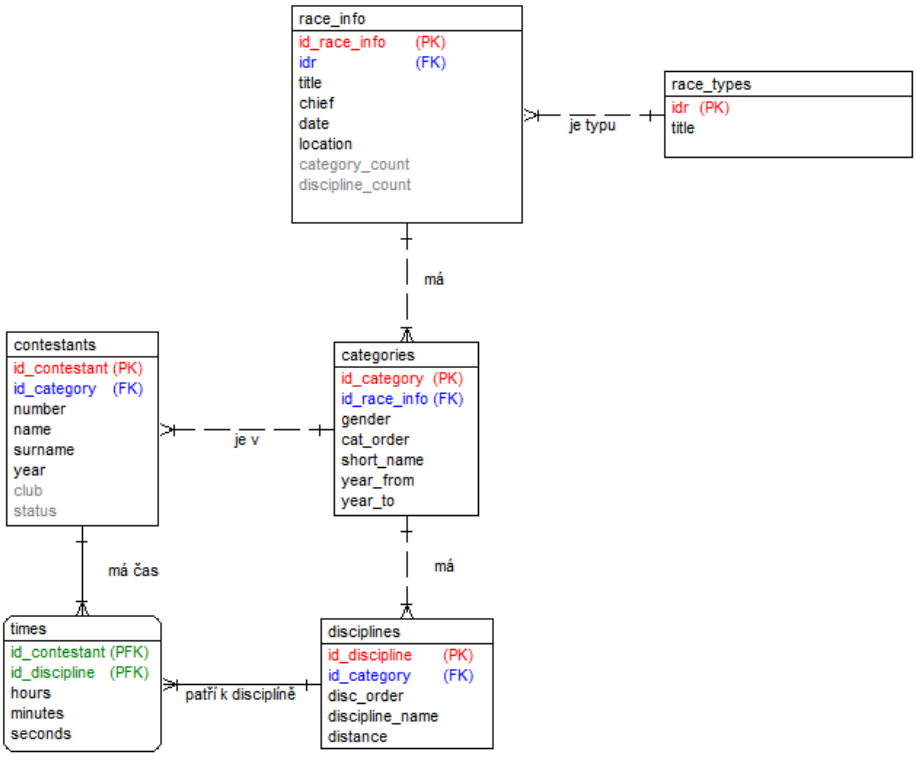
Mezi důležité informace o závodě patří název závodu, jméno pořadatele závodu a datum a místo uskutečnění. Zároveň je nutné zaznamenat informace o jednotlivých kategoriích závodu, tzn. zda-li je daná kategorie pro muže nebo ženy a jaký věkový rozsah tuto kategorii definuje. A poslední složkou jsou objemy (vzdálenosti), které jednotlivé kategorie musí zdolat pro absolvování závodu.

K závodníkům je třeba zjistit jejich startovní číslo, jméno a příjmení a ročník narození. Případně je možné zapsat i jejich sportovní oddíl, za který závodí.

U časomíry je nutné zaznamenat jak startovní číslo, tak disciplínu, ke které daný čas náleží.

2.2 Datová vrstva

Rozvrhnutí databáze jsem tvořil několikrát v průběhu vývoje aplikace, jelikož jsem vždy zjistil, že mi některé vazby vyhovují jinak a některé prvky chybí. Nakonec jsem vytvořil tuto strukturu, která mým účelům vyhovovala.



Obrázek 9: Databázová struktura

V obrázku č. 9 jsou znázorněny vazby mezi jednotlivými tabulkami. Nejdůležitější pro cílovou funkci programu jsou tabulky „contestants“ (závodníci) a „times“ (časy), které v sobě uchovávají, jaké startovní číslo (a tím pádem i který závodník) má jaký čas v dané disciplíně. To je pak třeba propojit s kategoriemi, aby se výsledky daly podle nich rozdělit. Každá kategorie se liší svou zkratkou (například benjamínci mají obvykle Benci) a věkovým rozmezím, podle kterého se jednotliví závodníci rozdělují do kategorií.

2.3 Aplikační vrstva

Samotnou práci s daty provádí kombinace PHP a JavaScriptu. Aplikace má takovéto rozložení:

index.html Jedná se o základní stavební blok, kam se vypisují veškeré akce aplikace. Zde se volá inicializace pomocí API souboru.

API Tento adresář obsahuje soubor index.php, který má za úkol sloučit všechny potřebné JavaScriptové soubory a kaskádové styly a vepsat je do kódu hlavní stránky.

Source V uvedeném adresáři se vyskytují všechny ovládací prvky offline webové aplikace.

- applicationcotroller.js - je nejdůležitejší, jelikož představuje mozek aplikace, který ovládá jaké funkce se mají volat a co se má zobrazit
- templates.js - udává jak mají vypadat jednotlivé stránky, které jsou pomocí HTML kódu napsány jako předlohy
- database.js - stará se o ukládání a načítání dat, jehož funkce volá controller

CSS Zde se nachází global.css, což je soubor obsahující kaskádové styly. Ty určují, jak jednotlivé HTML elementy mají vypadat (velikost, zbarvení atd.).

Backend Zde se nachází serverová část pro nahrávání dat.

- upload.php - jediný přístupný soubor užívateli, který se chová jako prostředník mezi uživatelem a PHP kódem serveru
- database.php - hlavním soubor, který se připojí k webové databázi a načítá nebo ukládá data v závislosti na volaných funkcích

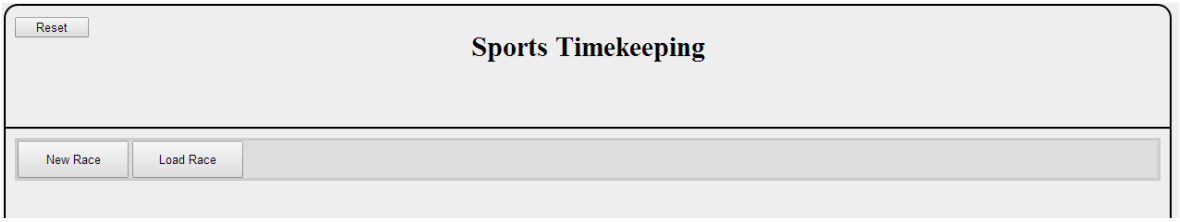
2.4 Prezentační vrstva

Po zvážení cílené funkce aplikace jsem zvolil jednoduchý a nezatěžující vzhled.

2.4.1 Počáteční stav

Při spuštění aplikace se uživateli zobrazí dvě základní možnosti. Vytvořit nový závod, nebo načíst již existující z offline databáze viz. obrázek č. 10.

Důvodem pro tuto volbu je, že můžeme chtít ještě před závodem uložit potřebné informace do databáze a následně zavřít prohlížeč (v horším případě může prohlížeč přestat fungovat a zavřít se sám). Tedy je nutné mít možnost znovu daný záznam otevřít.

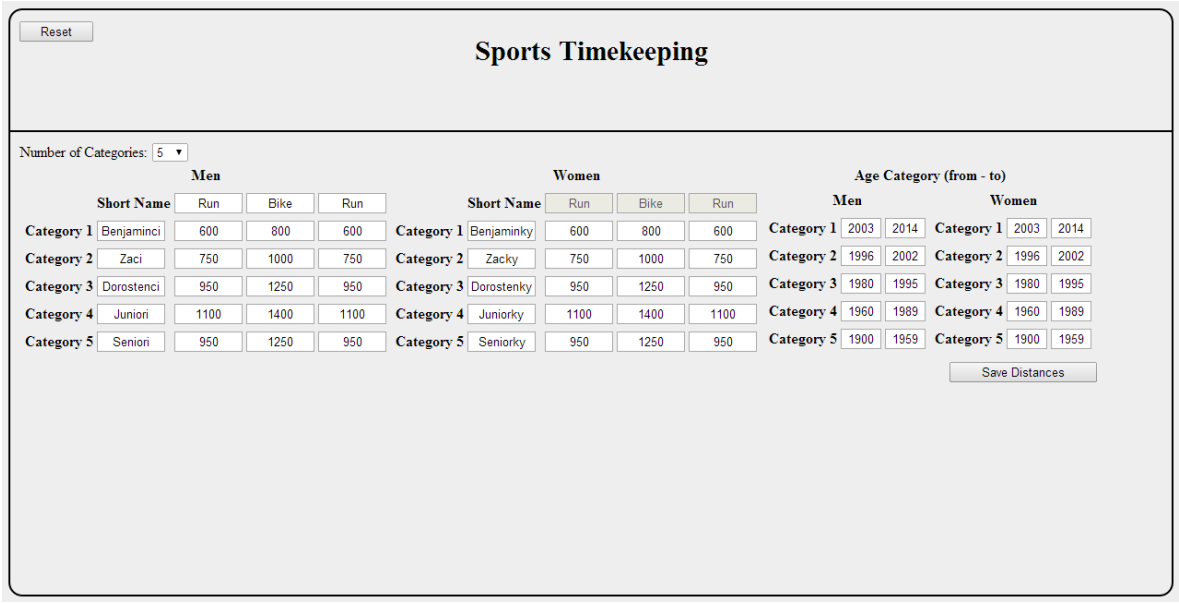


Obrázek 10: Start aplikace

2.4.2 Zadávání kategorií

Během vytváření nového závodu je třeba zadat parametry závodu, tedy počet kategorií, vzdálenosti jejich disciplín a věkové rozmezí viz. obrázek č. 11.

Uvedená část je velmi důležitá, jelikož aplikace následně podle těchto parametrů rozděluje závodníky do stanovených kategorií, což ovlivní, kteří závodníci budou proti sobě porovnáváni ve výsledkové listině.



Obrázek 11: Zadávání kategorií

V ukázce č. 11 je vytvářen závod typu triatlon, který má tři disciplíny (obvykle běh, kolo, běh). Je zvoleno pět kategorií a jak je vidět, každá kategorie se může pojmenovat dle potřeby (v ukázce jsou nejběžnější kategorie), nastavit jednotlivé vzdálenosti disciplín a určit věkový interval, který určuje danou kategorii.

2.4.3 Hlavní menu

Mezi základní funkce, které jsou uživateli k dispozici z nabídky, patří:

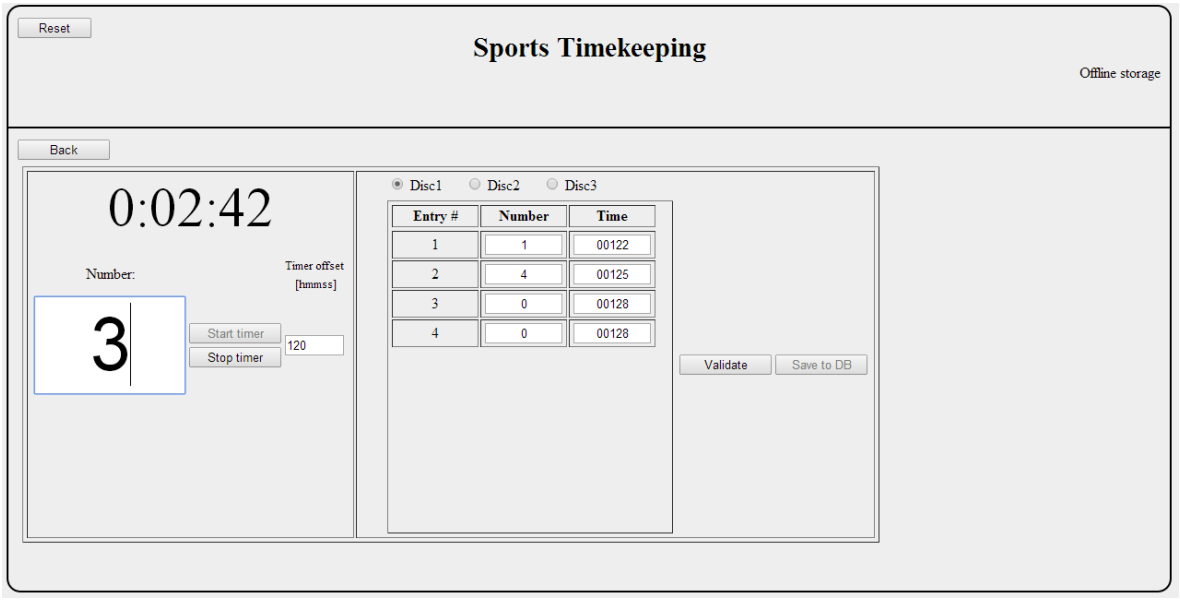
- Změna parametrů závodu (kategorie a jejich věkové ohraničení)
- Přidání nového závodníka
- Zobrazení startovní listiny
- Rozhraní pro měření a zaznamenávání časů
- Zobrazení výsledkové listiny
- Editace jednotlivých závodníků

Obrázek 12: Hlavní menu

V obrázku č. 12 je vidět hlavní nabídka, která je popsána výše. Jednotlivé položky jsou popsány zleva doprava. Dále má uživatel možnost vyresetovat aplikaci, čímž se dostane zpět do počátečního stavu (viz. obrázek č. 10). A při najetí na nápis „Offline Storage“ se zpřístupní možnost nahrát data do online databáze.

2.4.4 Zadávání časů

Tato část je nejdůležitější část programu, jelikož uživateli umožňuje spustit časomíru a přiřazovat časy k číslům závodníků.



Obrázek 13: Časomíra

Na obrázku č. 13 je zobrazeno rozmístění jednotlivých prvků pro zaznamenávání časů. Po zmáčknutí tlačítka „Start timer“ se spustí čas (zde má uživatel možnost nastavit časovou předvolbu, která spustí čas od zadané hodnoty). Poté již stačí do zadávacího pole napsat číslo (např. přijíždějícího) závodníka a dát enter. Program si pak uvedené údaje zaznamená do tabulky vedle. Tabulku lze editovat pro případ, že se uživatel zmýlí a zadá špatné číslo, nebo třeba zmáčkne enter moc brzo.

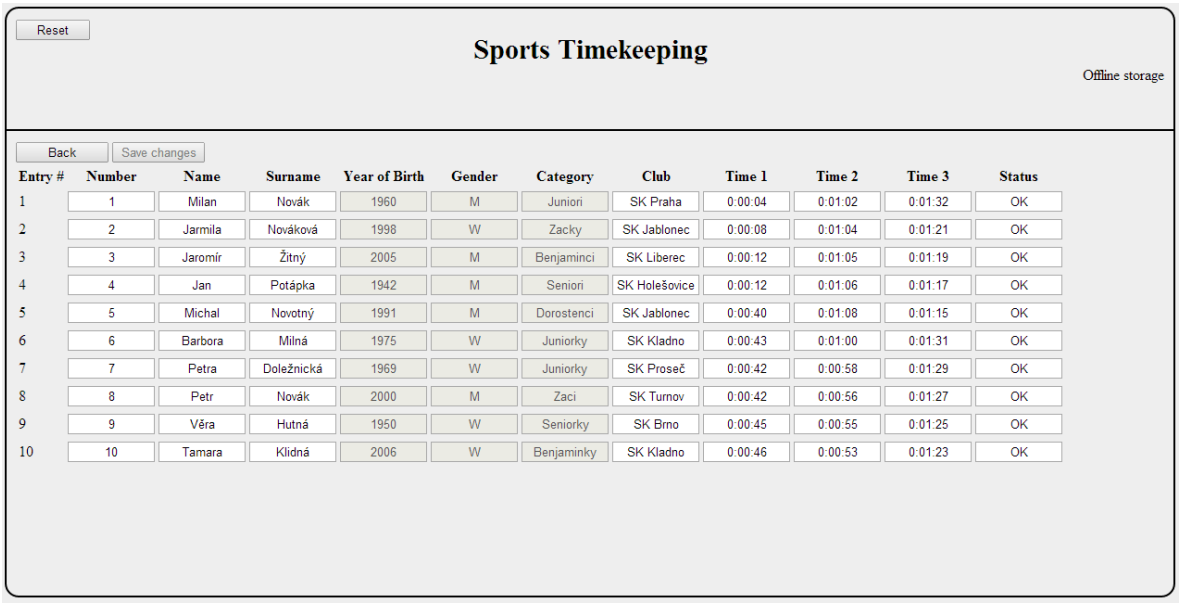
Program také umožňuje uložit čas bez nutnosti zadat číslo. To se pak projeví jako číslo 0 v tabulce. Tato možnost je velmi užitečná, když se například do cíle blíží větší počet závodníků, tak uživatel může jen sledovat jak kdo dokončil a až následně dopsat čísla (poznámka ze zkušenosti, v těchto případech si další člověk píše pořadí čísel v tomto shluku, takže uživatel aplikace neztratí žádné důležité informace).

Nad zmíněnou tabulkou lze vidět přepínání mezi jednotlivými disciplínami, jelikož každá disciplína může být měřena samostatně (tzv. mezičasy, které určují jak dlouho závodník strávil na dané disciplíně).

Po skončení, nebo i během závodu, může uživatel zvalidovat a uložit tabulku do databáze. Pod pojmem validace mám na mysli, že program zkontroluje zda-li v tabulce nejsou duplicitní čísla, nebo nepřiřazené časy, označené číslem 0.

2.4.5 Editace závodníků

Aplikace umožňuje změnit závodní údaje u závodníků v jedné přehledné tabulce viz. obrázek č. 14. Může se stát, že během závodu došlo ke komplikacím, jako například že si dva závodníci omylem prohodili startovní čísla, nebo závodník byl nějakých důvodů diskvalifikován. V takovýchto případech je nutné přepsat záznamy v databázi, aby odpovídaly skutečnosti. Jak



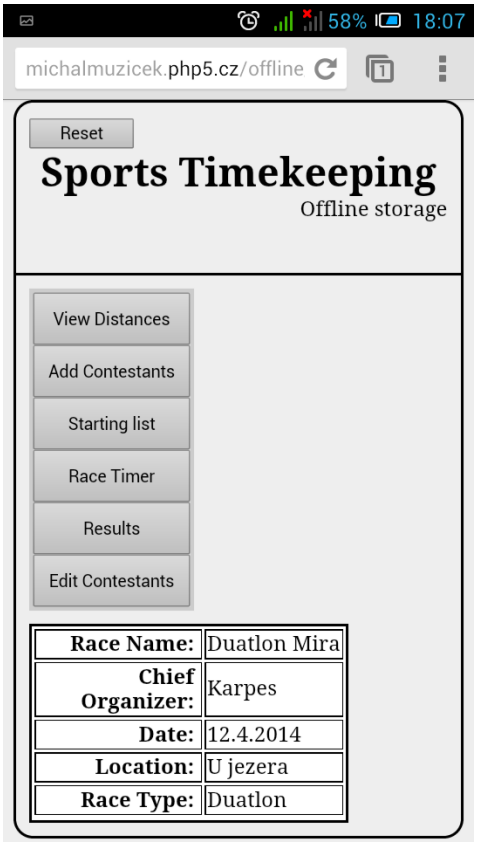
Entry #	Number	Name	Surname	Year of Birth	Gender	Category	Club	Time 1	Time 2	Time 3	Status
1	1	Milan	Novák	1960	M	Juniori	SK Praha	0:00:04	0:01:02	0:01:32	OK
2	2	Jarmila	Nováková	1998	W	Zacky	SK Jablonec	0:00:08	0:01:04	0:01:21	OK
3	3	Jaromír	Žitný	2005	M	Benjaminci	SK Liberec	0:00:12	0:01:05	0:01:19	OK
4	4	Jan	Potápka	1942	M	Seniori	SK Holešovice	0:00:12	0:01:06	0:01:17	OK
5	5	Michal	Novotný	1991	M	Dorostenci	SK Jablonec	0:00:40	0:01:08	0:01:15	OK
6	6	Barbora	Milná	1975	W	Juniorky	SK Kladno	0:00:43	0:01:00	0:01:31	OK
7	7	Petra	Doležnická	1969	W	Juniorky	SK Proseč	0:00:42	0:00:58	0:01:29	OK
8	8	Petr	Novák	2000	M	Zaci	SK Turnov	0:00:42	0:00:56	0:01:27	OK
9	9	Věra	Hutná	1950	W	Seniorky	SK Brno	0:00:45	0:00:55	0:01:25	OK
10	10	Tamara	Klidná	2006	W	Benjaminky	SK Kladno	0:00:46	0:00:53	0:01:23	OK

Obrázek 14: Editace závodníků

je vidět z obrázku č. 14, je možné nastavit libovolně časy jednotlivých disciplín. Aplikace je nastavena tak, aby zpřístupnila možnost uložení až poté co se nějaký element změní, zároveň ukládací tlačítko zajistí, že uživatel neuloží omylem změny, které nechtěl udělat (jelikož některé aplikace mohou být řešeny pomocí automatického ukládání při změně).

2.5 Responsive design

Aplikace je navržená tak, aby při menším displeji (chytré telefony mají obvykle šířku 400 pixelů) přeuspořádala ovládací prvky pro pohodlnější ovládaní. Jedná se hlavně o základní menu, kde se jednotlivá tlačítka zarovnají pod sebe, čímž je možné je všechny zobrazit na obrazovku viz. obrázek č. 15.



Obrázek 15: Rensponsive Design

3 Architektura aplikace

Při zvažování komponent k použití v aplikaci jsem měl několik možností.

3.1 Cílový prohlížeč

Jelikož se jedná o webovou aplikaci, tak přijde zřejmé, že by měla jít použít ve všech prohlížečích, avšak tato aplikace je postavena na prvcích standardu HTML5, který je relativně nový a nemá zatím širokou (a unifikovanou) podporu. Vybral jsem Google Chrome jako cílový prohlížeč, jelikož podporuje asynchronní ukládání dat a již jsem na práci s ním zvyklý.

3.2 Režim ukládání dat

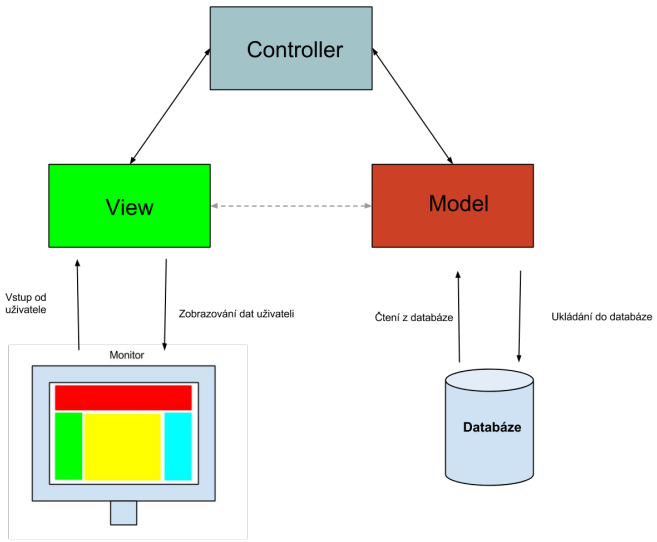
Vybral jsem asynchronní režim ukládání, neboť aplikace pracuje s obecně velkým množstvím dat a je třeba, aby se mohlo provádět několik operací najednou.

3.3 Webové prvky

Aplikace používá klasické prvky webových aplikací. Tedy HTML ,CSS ,PHP a JavaScript. Jelikož hlavní část programu se vykonává na části klienta (v prohlížeči), tak nepoužívám AJAX (asynchronní komunikace se serverem pomocí JavaScriptu)

3.4 MVC architektura

Aplikace byla navržena dle modelu MVC, který byl popsán výše a používá pouze jeden modul typu View, jelikož v jakýmkoli momentě zobrazuje daná data pouze v jedné podobě.



Obrázek 16: Schéma MVC modelu

V obrázku č. 16 je tedy vidět, že View zobrazuje uživateli data získané z databáze. V mém případě, jak bylo řečeno, mám pouze jeden modul View (v případě, že bych chtěl nějaká data zobrazit více způsoby, tak bych jich použil více...například kdyby bylo potřeba zobrazit časy závodníků v seřazené tabulce a zároveň tytéž data použít v grafu shrnující výkon závodníků)

3.5 Online databáze

Problematiku volby cílové webové databáze jsem vyřešil použitím bezplatného webového hostingu www.php5.cz, čímž jsem si udal formát SQL příkazů (tento hosting používá MySQL5).

3.6 Synchronizace databáze

Aplikace je nastavena tak, aby při nahrávání dat na server porovnala unikátní parametry závodu (žádný závod se nebude jmenovat stejně, konat se ve stejný den, mít stejného vedoucího atd.) a tím zjistila, zda se na webové databázi již daný závod nenachází. V případě shody pak nahradí webový obsah tím lokáním, jelikož z charakteru aplikace budou na lokáním úložišti vždy ta nejaktuálnější data.

3.7 Synchronizace mezi více uživateli

Jelikož aplikace je schopná nahrát své údaje pouze na jednu konkrétní databázi (na webu, na kterém je aplikace nahraná), vzniká problém konzistence dat když více aplikací nahraje svá data na server. To jsem vyřešil přidáním pomocného klíče, který zajistí integritu dat. Uvedeného je docíleno pomocí spolupráce pomocného klíče a algoritmu pro zjištění, zda daný závod již není v databázi uložen.

Při nahrání dat na webovou databázi tak proběhne synchronizace dat závodu v lokální databázi a dat závodu na webu (s tím, že z principu aplikace na lokální databázi bude vždy ta nejaktuálnější verze dat).

3.8 Testování funkčnosti v terénu

Během testování aplikace jsem vyzkoušel předem nastavit parametry závodu a vypnout aplikaci. Po znovuspuštění jsem zadal běžný počet závodníků (vzhledem k cílenému druhu závodů) a postupně zadával časy dle zkušeností z předchozích závodů. Program byl schopen zpracovat výsledky tak, jak bylo očekáváno a následně je nahrát na online databázi.

V rámci testování jsem změnil některé údaje v offline databázi a znovu nahrál. Na webové databázi se data správně synchronizovala.

Aplikace obstála i při testu nahrávání jiných závodů z jiných počítačů. Jelikož aplikace při nahrávání porovnává unikátní parametry závodu (žádný závod se nebude stejně jmenovat, mít stejného vedoucího, konat na stejném místě a ve stejném datu), tak se vždy synchronizují pouze ta data, které k sobě patří.

4 Závěr

V rámci své bakalářské práce jsem se seznámil s novými prvky, které přinesl standard HTML5 a využil jejich vlastností pro tvorbu své aplikace. Díky tomu může moje práce sloužit jako podklad pro kohokoli, kdo by chtěl využít tohoto standardu ve prospěch své webové aplikace.

V mém původním návrhu měla být tato aplikace jako desktopová. Ale po zjištění, že nové prvky v HTML5 umožňují pracovat s webovou aplikací i bez připojení k internetu jsem se rozhodl pro webovou aplikaci, jelikož je univerzálnější co se operačního systému týče a momentální směr informačních technologií se uchyluje spíše k oblasti internetu. Velikou výhodou webových aplikací je, že uživatel nemusí instalovat žádné podpůrné knihovny. Stačí, aby měl prohlížeč a ten už vše potřebné zajistí, což je jeden z dalších důvodů, proč jsem si tento typ aplikace vybral.

Po seznámení s prvky standardu HTML5 jsem se rozhodoval jakým stylem budu ukládat offline data a zvolil jsem asynchronní databázi. Poté jsem vybíral mezi použitím prohlížeče Mozzila Firefox nebo Google Chrome a po zvážení jejich kladů a záporů jsem zvolil Chrome. Po zvážení a následně navrhl a vytvořil offline webovou aplikaci tak, aby byla použitelná i na „chytrých“ telefonech pomocí technologie Responsive Design. Testování aplikace v terénu proběhlo bez problému a aplikace vykonávala přesně to, co budu potřebovat při zpracovávání výsledků při závodech.

Při tvorbě jsem zvolil Google Chrome jako cílový prohlížeč, jelikož podporoval asynchronní styl ukládání offline dat a svými atributy mi vyhovoval nejvíce. Alternativně by jistě mohly být použity ostatní prohlížeče, ale bylo by třeba dát pozor na jejich odlišné vlastnosti. Některé například nepodporují úložný model databáze a celá aplikace by se pak musela ukládat do proměnných, což by značně změnilo charakter kódu (nehledě na fakt, že by práce tohoto typu byla prováděna synchronně, tedy jednotlivé operace by čekaly, než se provede ta aktuální).

Aplikace by se dala do budoucna rozšířit například i o automatickou obsluhu webového obsahu, neboť aplikace pouze nahraje data na webovou databázi. V rámci spravování závodů by bylo pohodlné mít možnost nahrát data a následně nechat automaticky vygenerovat webový článek popisující daný závod včetně tabulky výsledků.

Použitá literatura

[1] Mark Pilgrim. HTML5: Up and Running. August 24, 2010. ISBN-10: 0596806027. ISBN-13: 978-0596806026

[2] Thomas M. Connoly. Carolyn E. Begg. Database Systems: A Practical Approach to Design, Implementation and Management. March 6, 2009. ISBN-10: 0321523067. ISBN-13: 978-0321523068

[3] Co to je webová aplikace. *Webová aplikace* [online]. January 31, 2014 [cit. 2014-01-31]. Dostupné z: `https://support.google.com/chrome/answer/1050586?hl=cs`

[4] Ukázka těla manifestu. *Tělo manifestu* [online]. January 31, 2014 [cit. 2014-01-31]. Dostupné z: `http://interval.cz/clanky/html-5-offline-webove-aplikace-a-aplikacni-cache/`

[5] Jednoduchý příklad. *Jednoduchý příklad* [online]. January 31, 2014 [cit. 2014-01-31]. Dostupné z: `http://labs.ft.com/2012/08/basic-offline-html5-web-app/`

[6] Responsive design framework Foundation. *Foundation* [online]. January 31, 2014 [cit. 2014-01-31]. Dostupné z: `http://foundation.zurb.com/`

[7] Rozdíly definice offline stavu. *Definice offline stavu* [online]. September 28, 2013 [cit. 2014-02-01]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/API/NavigatorOnLine.onLine`

[8] Úvod do architektury MVC. *Architektura MVC* [online]. May 7, 2009 [cit. 2014-02-23]. Dostupné z: `http://www.zdrojak.cz/clanky/uvod-do-architektury-mvc/`

Obsah přiloženého CD

- text bakalářské práce
  - bakalarska_prace_2014_Michal_Muzicek.pdf
  - bakalarska_prace_2014_Michal_Muzicek.tex
- zdrojový kód programu
  - pro webový server (kód využívá jazyků HTML, PHP, CSS a JavaScript)